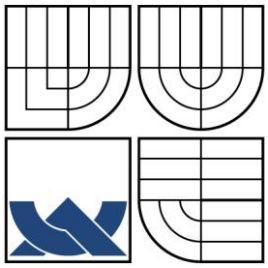
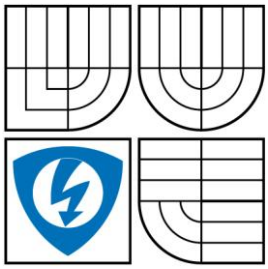


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SLEDOVÁNÍ ČÁRY PRO POŠTOVNÍHO ROBOTA

LINE TRACKING FOR DELIVERY ROBOT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MIROSLAV JUHAS
AUTHOR

VEDOUCÍ PRÁCE — Ing. ILONA KALOVÁ, Ph.D.
SUPERVISOR

BRNO 2008

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ
Fakulta elektrotechniky
a komunikačních technologií
Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Juhas Miroslav ***ID:*** 78566
Ročník: 3 ***Akademický rok:*** 2007/08

NÁZEV TÉMATU:

Sledování čáry pro poštovního robota

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte algoritmus pro detekci opticky zvýrazněné trajektorie v obraze a vyberte metodu vhodnou pro navigaci robota. Řešte i složitější případy např. měnící se světelné podmínky a vlastnosti sledované scény, zatáčky a křižovatky.
Předpokládá se praktická realizace v jazyce C/C++ s aplikací na reálného robota s pracovním názvem Pošťák, jehož úkolem je rozvoz dokumentů po UAMT.

DOPORUČENÁ LITERATURA:

Hlaváč, V., Šonka, M.: Počítačové vidění. Praha: Grada, 1992.

Termín zadání: 1.2.2008 ***Termín odevzdání:*** 2.6.2008

Vedoucí projektu: Ing. Ilona Kalová, Ph.D.

prof. Ing. Pavel Jura, CSc.
předseda oborové rady

UPOZORNĚNÍ:

Anotace

Práce popisuje základy metod a postupů v počítačovém vidění a také obsahuje návrh praktického uplatnění získaných poznatků na úloze - sledování čáry pro poštovního robota.

První část práce obsahuje základní teoretické poznatky z oboru počítačového vidění, které jsou nezbytné pro pochopení následujícího řešení, a je tedy úvodem do samotné podstaty problematiky.

V druhé části práce je uveden postup řešení. Je uvedeno samotné řešení jednotlivých kroků, kterými jsou předzpracování obrazu, segmentace, detekce trajektorie a vlastní algoritmus řízení směru pohybu, a zhodnoceny výsledky jednotlivých kroků. V závislosti na dosažených výsledcích jsou zde vybrány metody vhodné pro použití v řešené úloze. V souvislosti s prací jsou uvedeny i zkušenosti aplikace algoritmu na platformě UTAR.

V závěru jsou zhodnoceny a shrnuty výsledky dosažené během řešení jednotlivých bodů zadání.

Klíčová slova: počítačové vidění, sledování čáry, prahování, detekce hran, Houghova transformace

Brno University of Technology

Faculty of Electrical Engineering and Communication

Department of Control, Measurement and instrumentation

Line tracking for delivery robot

Thesis

Specialization of study: Computer vision

Student: Miroslav Juhas

Supervisor: Ing. Ilona Kalová, Ph.D.

Abstract:

This thesis describes basics of methods and algorithms used in computer vision and application of them at a simple practical problem – line-tracking for delivery robot.

The first part of this thesis contains basic theoretical knowledge of computer vision, which is important for understanding the problem. It is an introduction to problems of computer vision.

The second part of thesis describes solving of particular steps, which are image preprocessing, segmentation, trajectory detection and algorithms for direction control. It contains outcomes of particular steps and selection of methods acceptable for solving the problem. There are presented experiences with tests of algorithms on the UTAR platform in context of this work.

The last part of thesis is evaluating results taken during work.

Keywords: computer vision, line tracking, thresholding, edge detection, Hough transform

Bibliografická citace:

JUHAS, M. *Sledování čáry pro poštovního robota.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 53 s. Vedoucí bakalářské práce Ing. Ilona Kalová, Ph.D.

P r o h l á š e n í

„Prohlašuji, že svou bakalářskou práci na téma "Sledování čáry pro poštovního robota" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : Podpis:

P o d ě k o v á n í

Děkuji tímto vedoucí práce Ing. Iloně Kalové, Ph.D. za cenné připomínky a rady při vypracování bakalářské práce.

V Brně dne : Podpis:

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. ÚVOD

Počítačové vidění se dnes používá v širokém spektru průmyslových a dalších aplikací, ať už jde o výstupní kontrolu kvality, kontrola označení výrobků či jejich rozpoznání, automatické čtení značek automobilů, klasifikace terénu v mobilní robotice, mapování a podobně.

Úlohu zpracování obrazu v počítačovém vidění lze rozdělit do několika kroků (Hlaváč [1]):

1. Snímání, digitalizace a uložení obrazu
2. Předzpracování
3. Segmentace obrazu na objekty
4. Popis objektů
5. Porozumění obsahu obrazu (na nejvyšší úrovni zpracování obrazu)

Problematika počítačového vidění je v dnešní době již velice dobře zvládnutá a práce si neklade za cíl získání nových poznatků či rozšíření těch stávajících. Hlavním záměrem je získání základních znalostí spojených s problematikou počítačového vidění a snaha o jejich aplikaci na skutečný problém – sledování čáry.

Čára může pro robota představovat cestu vyznačenou v prostředí a s její pomocí může být schopen se v tomto prostředí autonomně pohybovat. Je tak možno zajistit navigaci robota mnohem jednodušeji, než kdyby se měl orientovat v obecném prostředí, bez vyznačené trajektorie. Využití takového robota může být, například, při střežení objektu, doručování výrobků, doručování pošty a další.

Cílem práce je vytvořit algoritmy pro sledování opticky zvýrazněné trajektorie, řešení zatáček, křižovatek a jejich značení.

2. ZÁKLADY POČÍTAČOVÉHO VIDĚNÍ

Kapitola popisuje pouze základy postupů zpracování obrazu v počítačovém vidění, a to pouze u vybraných částí problému. Větší část textu je čerpána ze zdroje (Hlaváč [1]). Pokud má čtenář hlubší zájem o problematiku mohu knihu „Hlaváč, V., Šonka, M.: Počítačové vidění“ doporučit jako výborný základ a úvod do problematiky počítačového vidění.

2.1 ZÍSKÁNÍ A REPREZENTACE OBRAZU

V současné době existuje několik způsobů získání obrazu. Jsou to například CCD kamery, skenery,…. Obraz bývá uložen a zpracováván v digitální podobě, což je z hlediska použití počítačů a digitální techniky výhodné, analogové snímání a zpracování obrazu se v dnešní době již téměř nepoužívá. Obraz může být reprezentován dvěma způsoby:

- Rastrová reprezentace – je reprezentací obrazu jako množiny bodů, každý bod obrazu je určen svojí polohou, řádkem a sloupcem v matici a také svojí barevnou hodnotou. Elektronické formáty JPEG, TIFF, RAW, PNG, BMP, GIF, WDP a další.
- Vektorová reprezentace – je reprezentací obrazu jako množiny objektů, např. přímek, kružnic,…. Standardizovaný elektronický formát SVG a dále DXF, VML, PLT, a další.

Jelikož počítačové vidění většinou pracuje s obrazem z reálné scény, který je získán technickými prostředky, jako jsou kamery apod., je obraz zpravidla množinou bodů, je tedy rastrový. Vektorová grafika se spíše používá v grafické tvorbě a pro základní aplikaci počítačového vidění nemá přílišný význam, proto se dále budeme zabývat pouze rastrovými obrazy.

Digitální obrazy bývají uloženy v různých formátech, tyto formáty jsou buďto bezztrátové (uložený obraz odpovídá původní předloze, klasickým představitelem je například formát raw, tiff nebo bmp) anebo ztrátové jako je jpeg apod. (jsou vypuštěna nadbytečná data, která nemají přílišný vliv na vzhled obrazu, ale jejich vypuštění umožňuje zmenšit datovou velikost obrazu).

Pro reprezentaci barvy každého bodu obrazu existuje několik barevných modelů:

1. Aditivní barevný model RGB – skládá se z jasů tří barevných složek: R-červená, G-zelená a B-modrá v rozsahu jasu 0-255.
2. Černobílý jasový model – skládá se z hodnot jasů 0-255.
3. Subtraktivní barevný model CMY (K) – je založen na principu odečítání barev (dochází k omezení spektra odraženého od povrchu), každá složka (C-azurová, M-purpurová, Y-žlutá) je reprezentována v rozsahu 0-100%.
4. Barevný model HSV – H-Hue (odstín), S-Saturation (sytost), V-Value (intenzita).

V počítačovém vidění se také často používá pouze dvoubarevný obraz (černá a bílá) pro segmentaci obrazu na objekty a pozadí.

Digitalizovaný obraz většinou bývá funkcí dvou argumentů (*i*,*j*) které vyjadřují polohu bodu. Hodnota funkce obrazu je jas bodu obrazu *g(i,j).*

2.2 PŘEDZPRACOVÁNÍ OBRAZU

Při aplikaci postupů počítačového vidění na získaný obraz je vhodné obraz nejprve předzpracovat, protože může obsahovat různá zkreslení a deformace vzniklé nedokonalým snímáním obrazu jako je například šum, neostrost, špatný jasový rozsah, geometrické zkreslení, zkreslení vzniklé nestejnoměrnou citlivostí snímače a další.

2.2.1 Jasové transformace

Dají se rozdělit do dvou skupin a to na jasové korekce (korekce závislé na poloze bodu obrazu) a modifikace jasové stupnice (nezávislé na poloze bodu obrazu).

Jasové korekce se používají pro korekce nestejnoměrného osvětlení nebo nestejnoměrné citlivosti snímače. Korekce je závislá na poloze bodu v obrazu. K výpočtu korekce se používá etalon, kterým je například stejnoměrně barevná plocha. Z rozdílu mezi etalonem a obrazem získaným snímacím zařízením se vypočítá korekční koeficient e pro každý bod (Hlaváč [1]):

$$f(i, j) = e(i, j)g(i, j) \tag{1}$$

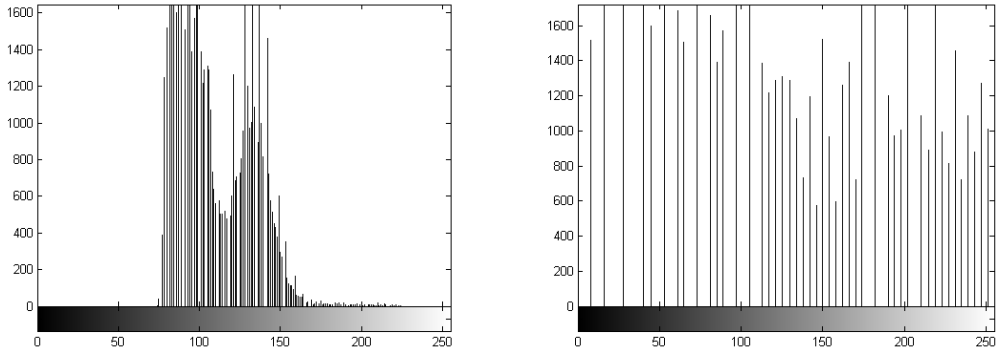
$$g(i,j) = \frac{f(i,j)}{e(i,j)} \tag{2}$$

Kde g je původní obraz, f je zkreslený obraz na výstupu zařízení a e je korekční koeficient.

Změna jasové stupnice se používá, například, pro zvýšení kontrastu obrazu, segmentaci prahováním, získání negativu a další. Každému bodu s určitým jasem je přiřazen jas nový, získaný z tabulky nebo podle matematické funkce. K této jasové transformaci se využívá histogramu. Histogram reprezentuje zastoupení jednotlivých jasů v obraze, na ose x je úroveň jasu, na ose y počet bodů, které každé úrovni jasu odpovídají. Nejčastěji se používá metoda ekvalizace (vyrovnání) histogramu, kdy je cílem, aby histogram pokrýval celou plochu grafu přibližně rovnoměrně. Výslednou jasovou transformaci pro diskrétní obraz hledáme ve tvaru (Hlaváč [1]):

$$q = T(p) = \frac{q_k - q_0}{N^2} \sum_{i=p_0}^{p} H(i) + q_0 \tag{3}$$

Obrázek 1: Obraz před a po ekvalizaci



Obrázek 2: Histogram před a po ekvalizaci

2.2.2 Geometrické transformace

Obraz získaný snímáním může být různým způsobem geometricky zkreslen. Toto zkreslení může vzniknout, například, vadou čočky, širokým ohniskem, jiným úhlem pohledu na scénu než kolmo, projekce 3D objektů do 2D prostoru apod. K získání správného obrazu používáme geometrické transformace, mezi ně patří, například, posunutí, změna měřítka, zkosení, dále se používají interpolace a registrace obrazu, jejichž cílem je geometrická rekonstrukce obrazu zjištěním takové transformace, která vede z daného souřadnicového systému do správného, k výpočtu se využívá klíčových bodů shodných pro oba obrazy.

K převedení mezi obrazem zkresleným a správným se používá mapování, existují dva způsoby (Mudrová [3]):

- Dopředné mapování – procházíme vstupní (zkreslený) obraz a hledáme odpovídající body ve výstupním obraze.
- Zpětné mapování – procházíme výstupní obraz a hledáme odpovídající body ve vstupním obraze.

Běžnými transformacemi obrazu jsou (Mudrová [3]):

- Lineární konformační - posunutí, otáčení, změna měřítka. Přímky zůstávají přímkami, úhly se nemění.
- Afinní - zkosení, přímky jsou zachovány, mění se úhly.
- Projektivní – změna rovnoběžek v různoběžky

- Polynomické – nový obraz je popsán křivkami (polynomy vyšších řádů)

2.2.3 Filtrace šumu

V digitálních snímacích zařízeních se většinou jako snímací prvek používá ccd snímač, tento snímač není dokonalý a při snímání na něm vzniká šum, jehož velikost je závislá na okolních podmínkách, jako je úroveň osvětlení, teplota… a také na vlastní nedokonalosti v konstrukci ccd čipu, nedokonalosti vzorkování obrazu a podobně. Šum většinou bývá v obraze rozložen náhodně a s náhodnými hodnotami jasu. U barevných ccd snímačů je zastoupení šumu různé pro různé barevné kanály, nejvíce šumu většinou obsahuje modrý kanál.

Odstranění šumu v obraze je vlastně proces, při kterém je obraz vyhlazován, neboli jsou z něj odstraňovány vysoké frekvence (rychlé změny hodnoty obrazové funkce), nevýhodou většiny algoritmů pro diskrétní filtraci obrazu je jeho rozmazání, což vede k potlačení hran. Hrany jsou však velice často používaným základem při segmentaci obrazu a dalších postupech v počítačovém vidění a jejich rozmazání bývá nežádoucí.

Základní metodou odstranění šumu, pokud máme k dispozici více obrazů snímané scény, bývá obyčejné průměrování. Pokud náhodné veličiny popisující aditivní šum jsou v jednotlivých bodech nezávislé a s nulovou směrodatnou odchylkou a střední hodnotou, je možno vypočítat jas každého bodu jako průměr hodnot z jednotlivých obrazů. Výhodou této metody filtrace je skutečnost, že nedochází k rozmazávání hran v obraze. Ne vždy máme k dispozici více obrazů snímané scény.

Pokud vycházíme pouze z jednoho obrazu, používáme lokální průměrování, kdy je hodnota jasu bodu vypočtena jako průměr hodnot jasu bodu a jeho okolí. Existuje mnoho masek pro lokální průměrování, většinou se liší velkostí a vlivem jednotlivých bodů na výsledný jas. Velikost masky je také dána nejmenšími detaily, které je nutné v obrazu zachovat. Metody filtrace lokálním průměrováním mají nevýhodu v rozmazávání hran v obraze.

$$h=\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad h=\frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad h=\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Obrázek 3: Příklady různých masek pro lokální průměrování (Hlaváč [1])

Často používanou metodou, která snižuje rozmazávání hran, je filtrace pomocí mediánu. Medián pro diskrétní obrazovou funkci je možno definovat jako prostřední hodnotu řady hodnot jasu. Výsledný jas je pak určen jako prostřední hodnota jasu okolních bodů. Používají se různé tvary a velikosti masek.

Z dalších metod se používá například rotující maska, kdy se hledá část okolí bodu, do kterého bod nejspíše patří, a poté je průměrná hodnota jasu bodu spočítána právě z tohoto okolí.

2.3 SEGMENTACE OBRAZU

Segmentace obrazu se zabývá hledáním těch oblastí v obraze, které představují objekty, nebo části obrazu, důležité pro pozdější zpracování. Při segmentaci se velice často využívá znalosti řešeného problému. Patří sem segmentace prahováním, detekce hran a z ní vycházející sledování hranice nebo Houghova transformace, dále pak segmentace narůstáním oblastí …. V této části bude věnována pozornost pouze základním metodám.

2.3.1 Prahování

Prahování využívá rozdílu mezi jasem pozadí a objekty ve scéně, hodí se pro segmentaci 2D obrazu, kdy objekty jsou od pozadí výrazně odlišeny jinou hodnotou jasu. Algoritmus prahování je jednoduchý, jas každého bodu obrazu je porovnán s hodnotou prahu, je-li jas bodu nižší, je mu přiřazena hodnota 0 (černá) neboli je označen jako bod objektu, je-li hodnota vyšší, je bodu přiřazena hodnota 1(bílá) neboli je označen jako bod pozadí. Reprezentace hodnot pozadí a objektu samozřejmě může být zvolena různě, v závislosti na řešeném problému nebo na vlastní volbě.

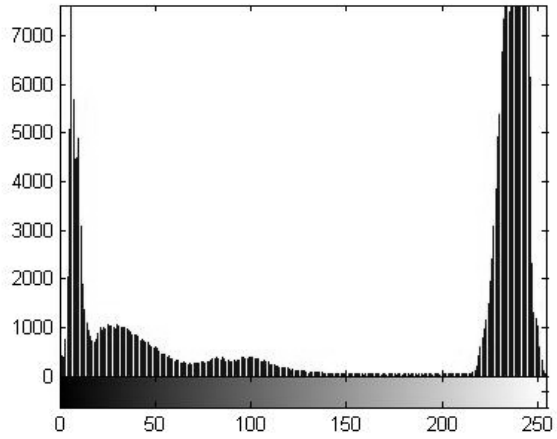
Výsledkem prahování je dvoubarevný obraz, který rozděluje scénu na objekty a pozadí. Nevýhodou je použitelnost pouze v případě jasového oddělení objektů od pozadí a nutnost správné volby prahu, aby bylo dosaženo správné segmentace.

Někdy se používá prahování s hysterezí, které do určité míry zlepšuje výsledky prahování. Je zavedena hystereze mezi hodnotou, kdy je přiřazena hodnota 1 a kdy 0. Pokud se jas bodu ocitne v pásmu hystereze, je mu přiřazena hodnota náležející objektu pouze tehdy, nachází-li se v jeho okolí jiný bod objektu, jinak je označen jako pozadí.

Obrázek 4: Prahování černobílého obrazu

Metody určení prahu jsou různé, většinou se hledá „údolí“ v histogramu, které rozděluje histogram na jasy náležící objektům a jasy náležící pozadí. Tato metoda je však závislá na znalosti histogramu a samotné nalezení správného minima v histogramu může být složité, navíc histogram ani nemusí takové minimum obsahovat, nebo může být výrazně zkreslen šumem. Existují i další metody určení prahu, které ani nemusí být závislé na znalosti histogramu obrazu. Jedna z těchto metod je součástí řešení zadání a bude tedy zmíněna v řešení zadané úlohy v další části práce.

Na následujícím obrázku, který zobrazuje histogram výše uvedené fotografie, je zřejmé, že nejlepší hodnota prahu bude někde mezi hodnotami 150 až 200.



Obrázek 5: Histogram pro prahování

2.3.2 Detektory hran

Další možností získání hranic objektů v obraze je detekce hran, tato metoda je výchozím bodem pro další segmentační metody, jako je sledování hranice nebo Houghova transformace.

Hrana v obraze je místo, kde dochází k velkému nárůstu hodnoty jasové funkce, zajímá nás tedy velikost a někdy i směr gradientu jasové funkce. U diskrétního obrazu se k získání hran využívá hranových operátorů, které je možno rozdělit do dvou skupin:

- Aproximace derivace jasové funkce pomocí diferencí, tzv. diskrétní konvoluce.
- Hledání hran v místech, kde druhá derivace jasové funkce prochází nulou.

Diskrétní konvoluce

Používá konvolučních jader, což jsou vlastně masky, které pomocí diference s různou vahou pro jednotlivé body počítají velikost hrany. Pro různé směry hran je nutno požívat různé natočení masky nebo se používá všesměrová maska. Diskrétní konvoluce se používá k ostření nebo detekci hran v obraze.

Po detekci hran také většinou následuje jejich prahování, abychom zvýraznili ty hrany, které jsou v obraze významné a jsou proto podstatné pro další zpracování.

Lze použít různá konvoluční jádra, která je pro jiné směry hran třeba otočit. Robertsův operátor je velmi jednoduchý a také velmi náchylný na šum.

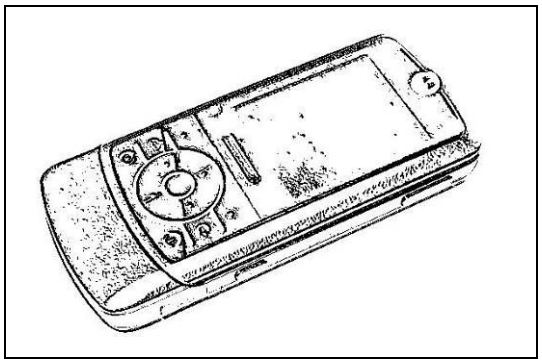
$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{4}$$

Sobelův operátor je lepší, ale pro každý směr je nutné jej vhodně pootočit.

$$h = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{5}$$

Existují i jádra, která jsou necitlivá na natočení, jako například Lapalceův gradientní operátor, ten se však používá k ostření obrazu.

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{6}$$



Obrázek 6: Obraz po aplikaci prahovaného Sobelova operátoru

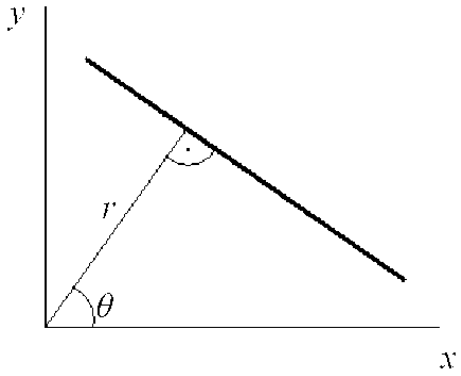
2.3.3 Houghova transformace

Houghova transformace je obecná metoda sloužící k hledání definovaných objektů v obraze. Protože vyžaduje, aby hledaný objekt byl parametricky popsán, klasická Houghova transformace slouží především k detekci útvarů, jako jsou úsečky, kružnice, elipsy atd. Zobecněná Houghova transformace však může být

použita také tam, kde není možný jednoduchý, analytický popis objektu.(Hořička [2])

Chceme-li najít například přímku, musíme zvolit její vhodný popis, většinou se používá popis přímky podle rovnice (7), protože parametry θ a r nejsou z celé množiny reálných čísel (θ je v intervalu <0,360), r má velikost ±úhlopříčka procházeného obrazu).

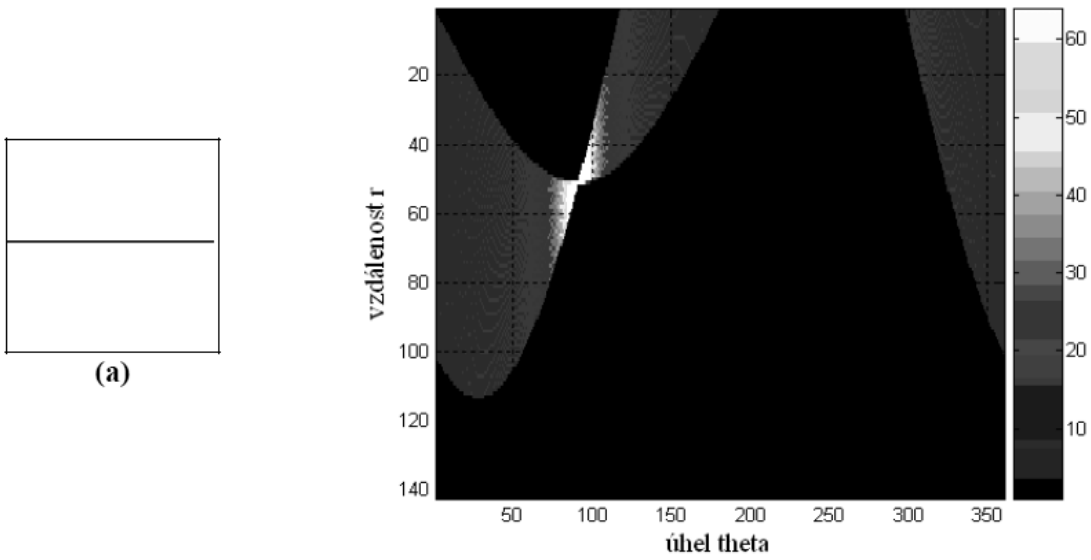
$$r = x\cos\theta + y\sin\theta \tag{7}$$



Obrázek 7: Reprezentace přímky

Pokud budeme v obrazu hledat přímku, vytvoříme nejdříve akumulátor, který bude obsahovat množinu všech hodnot parametrů r a θ. Tento akumulátor bývá reprezentován vícerozměrnou maticí v závislosti na počtu parametrů, v našem případě dvourozměrnou. Na počátku jsou všechny hodnoty nulové.

Při aplikaci transformace procházíme obraz bod po bodu, narazíme-li na bod (souřadnice bodu jsou x a y) náležející předmětu (přímce), jsou vypočteny všechny hodnoty r pro všechna θ, pro každou dvojici $[r,\theta]$ je inkrementována příslušná hodnota v akumulátoru. Když je takto analyzován celý obraz, je možno najít parametry přímky nalezením maxima v akumulátoru. Situace je jasně patrná z obrázku 8. Je samozřejmě možné hledat i více lokálních maxim a tím další přímky v obraze.



Obrázek 8: a) obraz vstupující do transformace b) obsah akumulátoru (Hořička[2])

2.4 DALŠÍ METODY POUŽÍVANÉ V POČÍTAČOVÉM VIDĚNÍ

- Matematická morfologie – například dilatace, eroze, otevření, uzavření, podmíněná dilatace a konečné eroze, skelet apod.
- Segmentace narůstáním oblastí
- Segmentace srovnávání se vzorem
- Různé druhy transformací
- Popis objektů a oblastí a jejich identifikace v obraze
- Klasifikace a automatické třídění
- Analýza pohybu (optický tok a detekce významných bodů)
- Analýza trojrozměrných objektů a další

3. NÁVRH POSTUPU ŘEŠENÍ

Celý postup řešení lze shrnout do 4 následujících kroků:

1. Návrh možných způsobů předzpracování obrazových dat a výběr nutných kroků v předzpracování obrazu.
2. Vyzkoušení možných metod částečné segmentace obrazu a výběr té nejvhodnější.
3. Volba vhodné metody pro nalezení objektu, jímž je zvýrazněná trajektorie, a matematické reprezentace této trajektorie, tedy vytvoření popisu objektu.
4. Vytvoření algoritmů, jejichž výstup udává směr, kterým se má robot pohybovat, řešení křižovatek a jejich značení.

V následujících kapitolách je uveden postup výběru jednotlivých kroků nutných ke zpracování obrazu a popis vytvořených algoritmů zajišťujících sledování trajektorie. Pro řešení problému byl v počátku využit program Matlab, protože již obsahuje toolboxy pro práci s obrazem. Později byla vytvořena aplikace v jazyce C++, pomocí Borland C++ Builder, použitá pro řízení podvozku platformy UTAR pomocí videokamery a rozhraní RS232. Zkušenosti s „nasazením v terénu“ jsou uváděny v kontextu s jednotlivými částmi práce a také shrnuty v kapitole 9. Aplikace v C++ je přílohou práce, stejně jako fotografie a videa z praktických testů.

Z důvodu tisku jsou všechny binární obrazy barevně invertovány tak, aby byly eliminovány velké černé plochy.

3.1 VLASTNOSTI OPTICKY ZVÝRAZNĚNÉ TRAJEKTORIE

Vzhled, vlastnosti a pravidla pro vlastnosti opticky zvýrazněné trajektorie (dále jen čáry) jsou vlastně jedním z výsledků této práce. Uvést je na počátku práce se však jeví výhodnější z hlediska pochopení toho, co se v obrazu snažíme následujícími algoritmy najít, popsat a podle čeho se chceme řídit. Definujme tedy vlastnosti čáry a sledované scény:

- Robot se pohybuje uvnitř budovy s umělým osvětlením nebo s osvětlením venkovním světlem.

- Jako čára byla zvolena barevná páska (červená, zelená nebo modrá). Čára musí být dobře odlišitelná od okolního prostředí a musí mít dostatečnou šířku, aby se dala v obraze rozpoznat. Touto čárou je definována cesta, po které se má robot pohybovat.
- Zatáčky jsou realizovány navazujícími přímkovými úseky, jedná se o kompromis mezi použitými algoritmy a podvozkem testovacího robota.
- Křižovatka je definována dvěma kolmými přímkami současně viditelnými kamerou.

Tyto podmínky vycházejí z algoritmů uvedených dále a za těchto podmínek byly tyto algoritmy testovány.

Obrázek 9: Příklad cesty, zatáčky a křižovatky

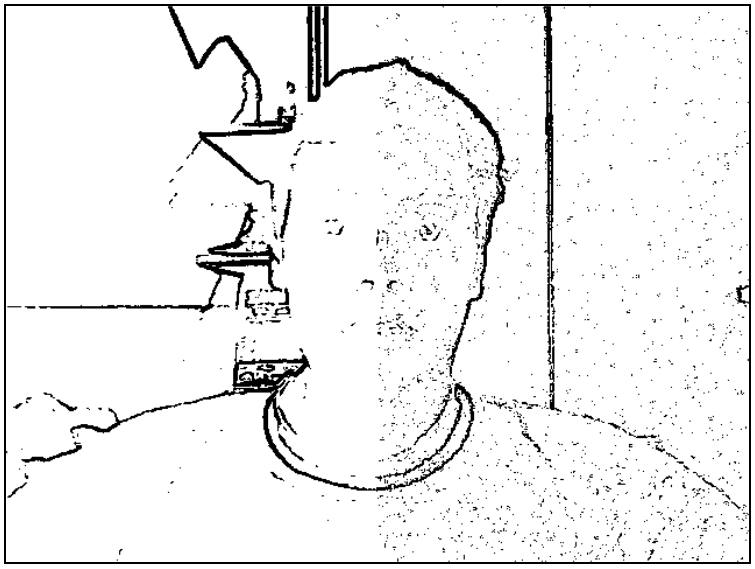
4. PŘEDZPRACOVÁNÍ OBRAZOVÝCH DAT

Je zřejmé, že obraz získaný reálnou kamerou bude trpět hned několika obrazovými vadami a to hlavně šumem, geometrickým zkreslením, pokud nebude čára snímána kolmo, a také je otázkou kontrastnost získaného obrazu.

4.1 REDUKCE ŠUMU

Při zkouškách s kamerou se ukázalo, že obraz je poměrně nekvalitní a je v něm zastoupeno poměrně velké množství šumu, který by překážel například při použití hranových operátorů, proto byla vyzkoušena filtrace šumu lokálním průměrováním při stejné váze všech bodů okolí, ta sice způsobuje rozmazání hran, ale jak se při zkouškách ukázalo, pokud je hrana výrazná je možno ji nadále detekovat některým z konvolučních jader. Vhodnější je použití filtrace mediánem s takovým tvarem masky, která nepotlačuje tenké hrany.

Jak ukazuje obrázek 10, vliv na prahovaný Sobelův operátor při detekci hran není příliš znatelný, i když k rozmazání hran došlo, vliv filtru na šum v obraze je zásadní, šum v pravé polovině je jasně znatelný a stěžoval by další práci s obrazem.



Obrázek 10: Vliv filtru šumu na Sobelův operátor 3x3

4.2 MÍCHÁNÍ BAREVNÝCH KANÁLŮ

Při popisu tohoto předzpracování obrazu je nutné předběhnout až k samotnému finálnímu testování na platformě UTAR. Původně byla, jako zvýrazněná trajektorie, uvažována černá čára na světlém podkladě, kterým je podlaha. Během testování se však ukázalo několik podstatných problémů z hlediska toho, jak zvýraznit trajektorii:

- Podlaha v prostorách UAMT, kde bylo testováno, není všude pouze světlá, ale vyskytují se například tmavé dlaždice.
- Budova, postavená člověkem, je prostředím, které obsahuje mnoho hran a kontrastních míst, které lze snadno zaměnit za čáru. Jsou to například spáry mezi dlaždicemi, lišty spojující linoleum, tmavé prahy, lišty u stěn a další.

Zatímco v prvním případě zklame vzhledem k malému kontrastu prahování, v druhém případě dostáváme při použití hranového detektoru velké množství silných hran i dvojic rovnoběžných hran, které není snadné odlišit od čáry, kterou chceme detekovat.

Vzhledem k těmto problémům bylo použito míchání barevných kanálů, při kterém bylo hlavním cílem zvýraznění trajektorie, kterou je v tomto případě barevná čára (červená, modrá nebo zelená).

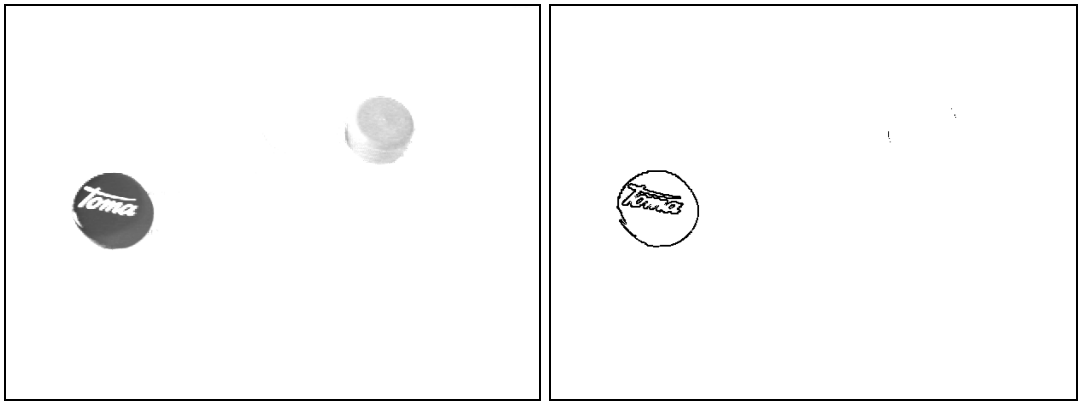
Pokud budeme chtít detekovat červenou čáru je nutné tuto barvu v obraze nějakým způsobem zvýraznit. Zajímají nás části obrazu, které obsahují z většiny barvu v červeném kanálu a téměř nejsou zastoupeny v ostatních barevných kanálech. Samotné využití pouze červeného kanálu není dostačující jak je vidět na obrázku 11. Mnohem lepším přístupem je využít toho, že je červený předmět vyjádřen vysokým jasem v červeném kanálu a v ostatních má nízký jas, zatímco předměty jiných barev mají vysoký jas v jiných, i více, kanálech. Pro výsledný jas bodů můžeme použít například výpočet:

$$JAS(i,j) = R(i,j) - (0.5\ G(i,j) + 0.5\ B(i,j)) \qquad (8)$$

Kde je ale nutno zajisti, aby když *JAS(i,j) < 0* pak *JAS(i,j)=0*. Výsledky tohoto míchání barevných kanálů jsou na následujících obrázcích.

Obrázek 11: Barevná testovací scéna a ukázka červeného kanálu[1]

Na obrázku 11 je vidět, že červený kanál příliš nepomůže při rozpoznání červeného objektu, ale dokáže vyřadit modrý a zelený objekt.

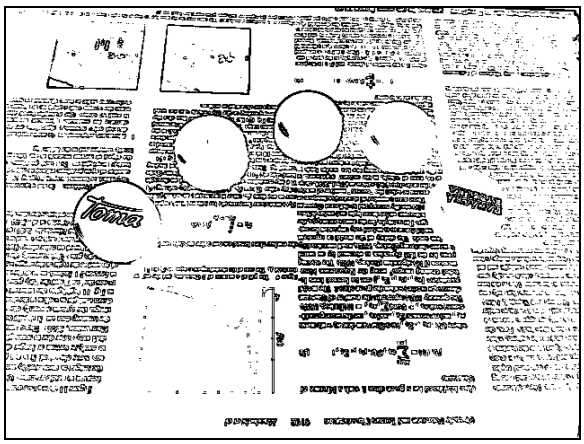


Obrázek 12: Výsledek aplikace vztahu (8) na barevný obraz a odpovídající prahovaný obraz hran

Jak je vidět na obrázku 12, byly potlačeny všechny předměty kromě červeného, oranžový není potlačen dokonale díky velkému podílu červené v odraženém barevném spektru, ale je možné jej oddělit prahovaným detektorem hran nebo prahováním. Důležité je, že byly potlačeny nejenom modré a zelené objekty, ale také ostatní objekty, kterými je například černý text na bílém pozadí.

[1] V případě černobílého tisku jsou barvy zleva doprava: červená, zelená, modrá, oranžová, žlutá

Došlo k tomu proto, že bílá i černá obsahuje přibližně podobně červené, modré i zelené v odraženém barevném spektru a po použití vztahu (8) vychází přibližně hodnota 0, což je černá. Porovnáme-li obrázky 12 a 13 zjistíme, že výsledek aplikace hranového detektoru je neporovnatelně lepší na obrázku 12, kde jsou hrany pouze červeného objektu.



Obrázek 13: Prahovaný obraz hran černobílého obrazu

Je nutné si uvědomit, že tento krok v předzpracování obrazu značně zjednodušuje následné zpracování, protože eliminuje velkou část rušivých vlivů (stíny, hrany předmětů, tmavé předměty,…), které by narušovaly detekci čáry.

V aplikaci později napsané v C++ je možné volit mezi černobílým obrazem nebo zvýrazněním kteréhokoliv barevného kanálu. Nejsme tak vázání na jeden vzhled čáry.

4.3 GEOMETRICKÉ KOREKCE

Hlavním geometrickým zkreslením, které by mohlo při snímání obrazu vzniknout, je lichoběžníkové, při pohledu kamery jinak než kolmo na podložku. Otázkou totiž je výška kamery, umístěné na robotu, a zda bude snímat čáru v dostatečné vzdálenosti před robotem, aby bylo možno reagovat na případné změny směru nebo křižovatky.

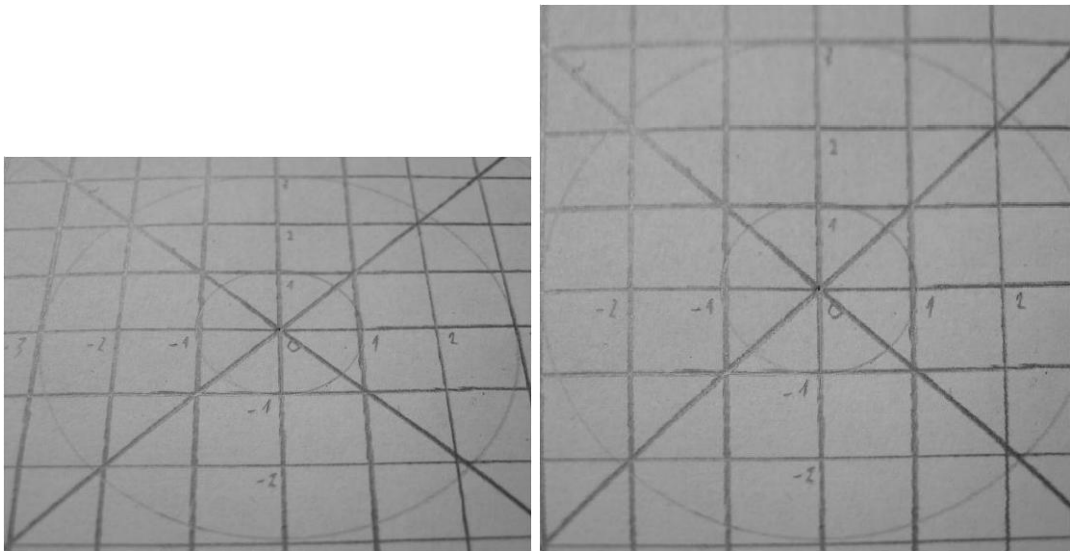
Pro korekci tohoto zkreslení je nutné použít projektivní transformaci (mění rovnoběžky v různoběžky). Projektivní transformace váže souřadné systémy podle vztahů:

$$x_1 = \frac{a_{11}u_1 + a_{12}u_2 + a_{13}}{a_{31}u_1 + a_{32}u_2 + a_{33}}$$

$$x_2 = \frac{a_{21}u_1 + a_{22}u_2 + a_{23}}{a_{31}u_1 + a_{32}u_2 + a_{33}} \qquad (9),(10)$$

Kde $[x_1, x_2]$ jsou souřadnice bodu v původním obrázku a $[u_1, u_2]$ jsou souřadnice bodu v novém obrázku.

Použitý algoritmus by měl používat zpětné mapování, kdy jsou nejdříve určeny rozměry výstupního obrazu a ten je pak procházen, přičemž jednotlivým bodům jsou přiřazeny odpovídající body z původního obrazu. Tím dojde k protažení obrazu, což odpovídá pohledu robota vpřed.



Obrázek 14: Geometrická korekce obrazu

Na výsledném obraze je vidět, že ke konci dochází k poměrně silné interpolaci, která by mohla silně degradovat kvalitu obrazu, pokud bychom požadovali takový úhel pohledu kamery, který by zabíral scénu daleko dopředu a také soudkovité zkreslení, které je způsobeno použitým objektivem, úhly v obraze se již blíží pravým úhlům a přímky jsou již rovnoběžné.

4.4 EKVALIZACE HISTOGRAMU

Nebyla vyzkoušena, z hlediska následných algoritmů jako jsou hranové operátory nebo prahování, není příliš podstatné, zda je například údolí při automatickém nastavení prahu detekováno z původního nebo ekvalizovaného histogramu. Výsledek bude v obou případech podobný a stejné body se dostanou pod práh nebo nad práh. Tato korekce má význam u obrazů, které mají být reprezentovány člověku.

4.5 VYBRANÉ ALGORITMY

Jako algoritmus, který bude použit při předzpracování, je dostačující filtr šumu. Vzhledem k možnosti segmentace detekcí hran je vhodnější filtrace mediánem s vhodným tvarem masky, nežli lokálním průměrováním.

Míchání barevných kanálů s cílem zvýraznit barevnou čáru je důležitým krokem, který zjednodušuje návrh algoritmů pro hledání trajektorie obrazu, protože výrazně snižuje množství objektů v obraze, které by mohly být považovány za danou trajektorii, a výrazně tím zjednodušuje segmentaci obrazu a rozpoznání čáry.

Ekvalizace histogramu pro zvýšení kontrastu scény není pro řešenou úlohu kritická, protože předpokládáme opticky zvýrazněnou trajektorii, jakou je například černá čára na bílém podkladu a podobně.

Geometrická korekce by měla svůj význam, nicméně vzhledem k algoritmům popsaným dále, kdy je pohled kamery příliš vpřed nežádoucí, nebyla využita a byla vyzkoušena pouze v prostředí programu Matlab.

Vybrané kroky při předzpracování obrazu jsou tedy:

- Filtrace šumu
- Míchání barevných kanálů

5. SEGMENTACE OBRAZU

Byly vyzkoušeny dva způsoby, prahování a detekce hran. Následné algoritmy k detekci čáry potřebují čáru jako bílý objekt, přičemž pozadí je černé. Je vcelku jedno, zda bude čára reprezentována jako jeden objekt vzniklý prahováním, nebo dva vzniklé detekcí hran čáry. Hlavní otázkou je, který postup segmentace je spolehlivější.

5.1 PRAHOVÁNÍ

Je nejjednodušší metodou segmentace, pokud zvolíme dokonale černou čáru na bílém podkladu, měl by být výsledkem obraz reprezentující hodnotou 1 čáru a hodnotou 0 její okolí.

Obrázek 15: Model čáry snímaný kamerou

Již na pokusném obrazu získaném kamerou je však zřejmé, že vše nebude tak snadné, jak se zdá. Obraz trpí nedostatečným osvětlením. Snímač v rozích obrazu zřetelně vykazuje menší citlivost, takže je zde obraz tmavší a to až tak, že velikost jasu v rozích obrazu se může blížit úrovni jasu čáry. Dalšími problémy, které mohou způsobit špatný výsledek segmentace, jsou například stíny, nerovnoměrné osvětlení a podobně.

Hodnota prahu nemůže být nastavena pevně, musí být automaticky měněna během chodu a to v závislosti na okolních podmínkách a tedy na kvalitě snímaného

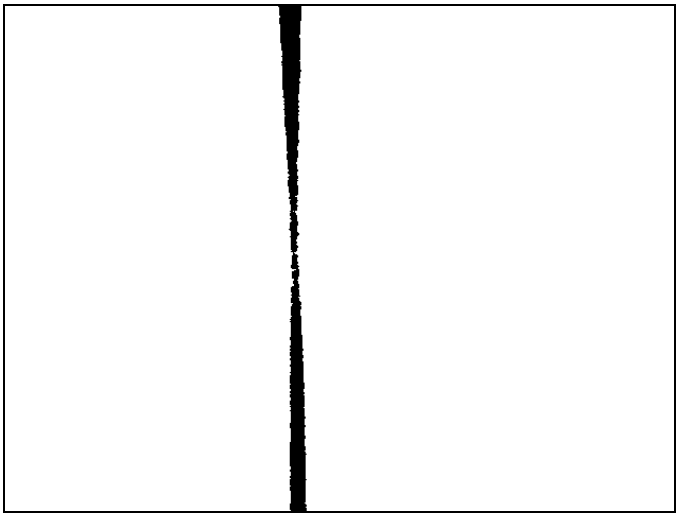
obrazu. Hledání údolí v histogramu může být poměrně složité, a někdy takové místo v jeho grafu ani nemusí existovat, proto byl použit algoritmus, který není závislý na znalosti histogramu a ani není příliš ovlivněn šumem obrazu. Jedná se o iterativní algoritmus, který dokáže již během několika snímků nastavit práh na správnou hodnotu (v rámci možností s ohledem na kvalitu obrazu).

Popis použitého algoritmu procentního prahování (Wikipedia [4]):

1. Je vybrán výchozí práh T, náhodně nebo přibližně, v závislosti na úloze.
2. Je provedeno prahování, přičemž se vytvoří dvě tabulky hodnot jasů – *G1* obsahující hodnoty jasů, které náležejí objektu a *G2* obsahující hodnoty jasů pozadí.
3. Je spočítána průměrná hodnota jasu každé tabulky (*m1=průměr G1,m2=průměr G2*).
4. Je vypočten nový práh jako $T=(m1+m2)/2$. Hodnota prahu nesmí klesnout pod hodnotu T_{min} , pokud $T<T_{min}$ pak $T=T_{min}$.
5. Celý algoritmus je opakován od kroku 2, dokud není dosaženo neměnnosti výsledného prahu (bylo dosaženo konvergence).

V našem případě, kdy je scéna snímána neustále, je i algoritmus neustále v běhu a počítá stále nový práh, díky tomu je možné reagovat například i na změny osvětlení a podobně. Hodnota minimálního prahu je použita z toho důvodu, že pokud robot například sjede z čáry, mohla by se hodnota prahu snížit natolik, že by se v prahovaném obraze objevily i velmi světlé objekty nebo šum. Mohlo by dojít k nalezení neexistující čáry a robot by ji začal sledovat, proto je nutné určit minimální hodnotu prahu.

Rychlost algoritmu se při testování ukázala jako poměrně dobrá a již během několika snímků byl výsledný obraz použitelný pro detekci čáry. Velikost výsledného prahu lze navíc ovlivnit změnou jmenovatele v kroku 5, pokud bychom například chtěli posunout práh výše, než je potřeba a tím dosáhnout zamezení detekce i některých objektů v okolí.



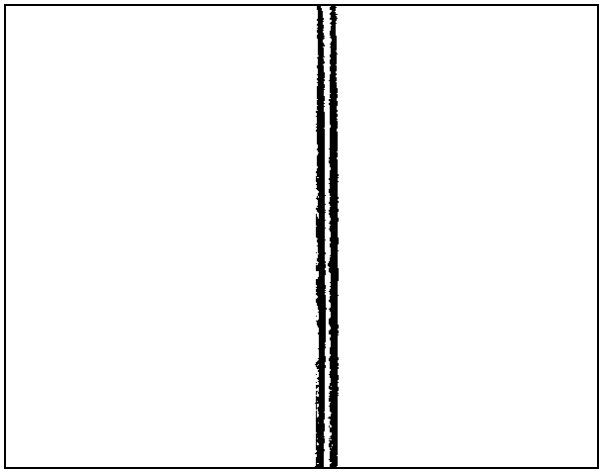
Obrázek 16: Obraz získaný procentním prahováním

5.2 DETEKCE HRAN

Jak už bylo uvedeno výše, pokud bude čára čistě černá na bílém pozadí, budou detekovány dvě hrany, jedna na každé straně čáry. Jako výhoda detekce hran se už předem jeví to, že není tolik citlivá na nestejnoměrné osvětlení. Pokud se osvětlení obrazu, nebo nestejnoměrná citlivost kamery, bude měnit v obraze spojitě, nedojde k detekci hrany a tím ani k detekci falešného objektu, což je výhoda oproti prahování. Nastavením prahování hrany na vysokou hodnotu lze vyřadit i hrany objektů, které by mohly do obrazu zasahovat, budeme-li předpokládat, že hrana čáry je velmi výrazná.

Faktorem, který by mohl „nabourat“ detekci hran, je například ostrý stín nebo jiná silná hrana poblíž čáry určené k navigaci robota, nebo její rozmazání vlivem pohybu robota.

V hranovém detektoru v projektu byl použit Sobelův operátor 3x3 pro vodorovný a svislý směr a procentní prahování s použitím algoritmu uvedeného v předcházejícím bodě aplikovaného na obraz hran.



Obrázek 17: Obraz získaný hranovým detektorem a procentním prahováním

5.3 VÝBĚR METODY SEGMENTACE

Během zkoušení obou způsobů segmentace, se ukázalo prahování jako nespolehlivé při špatném osvětlení a horší kvalitě obrazu. Naproti tomu detekce hran umožňuje jemnější nastavení prahu hran, nevýhodou však je větší výpočetní náročnost oproti prahování. Pokud je obraz horší a vyskytují se v něm tmavá místa bez rušivých hran, je lepší volit pro segmentace detekci hran. Pokud máme kvalitní obraz bez dalších tmavých rušivých objektů a jsme omezeni možnou výpočetní náročností, je lepší volit jednodušší prahování.

Vzhledem k tomu, že by se měl robot pohybovat po chodbách, kde mohou panovat různé světelné podmínky v závislosti na denní době a podobně, jeví se jako lepší algoritmus detekce hran. Bohužel během testů na platformě UTAR se ukázalo, že použitá kamera při pohybu robota produkuje pohybem rozmazaný obraz a dochází ke ztrátě mnoha hran. Rozmazání hran vedlo k nesprávné segmentaci a ztrátě některých informací důležitých pro navigaci robota, proto bylo při testech použito pouze prahování. Tento problém by bylo možné řešit lepší kamerou, která by pracovala s kratším časem expozice nebo některým filtrem odstraňujícím rozmazání obrazu (například Wienerův filtr). Všechny následující algoritmy, kromě čtení označení křižovatek, kde je nutná drobná úprava algoritmu, pracují stejně jak s prahovaným, tak s hranovaným obrazem.

6. DETEKCE ZVÝRAZNĚNÉ TRAJEKTORIE

K detekci a reprezentaci čáry byla na začátku zvolen poměrně jednoduchá metoda, později se ukázalo, že má svá omezení, hlavně co se týče situací, kdy jsou v obraze další čáry (např. křižovatky či rušivé signály). Jednoduchost metody vzhledem k těmto problémům byla spíše problémem než výhodou. Proto byl později zvolen jiný přístup jak k matematickému popisu zvýrazněné trajektorie, tak ke způsobu její detekce.

6.1 PŮVODNÍ METODA DETEKCE A REPREZENTACE TRAJEKTORIE

Byla zvolena matematická reprezentace přímky ve tvaru

$$y = kx + b \tag{11}$$

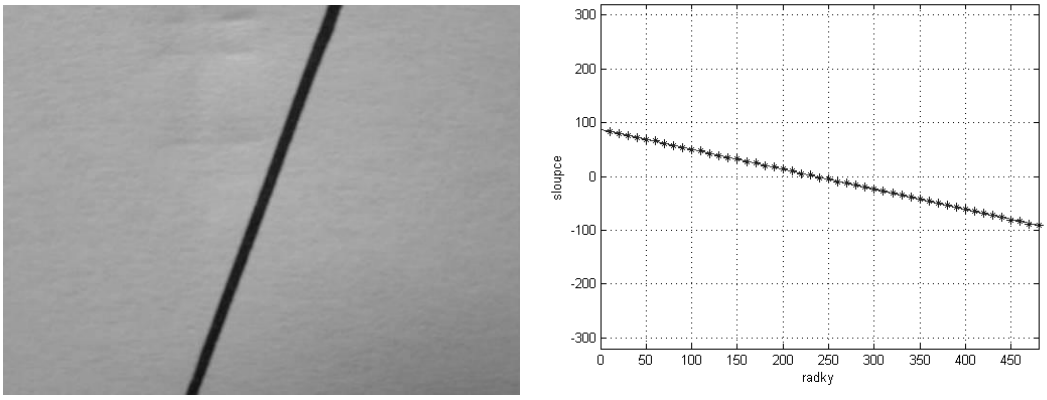
Bohužel v tomto tvaru není s dostatečnou přesností možno zobrazit přímky, jejichž směrnice se blíží nekonečnu, neboli přímky rovnoběžné s osou *y*, proto musely být realizovány dva souřadné systémy, jeden pro přímky se směrnicí $<< \infty$, a druhý pro přímky se směrnicí blížící se k ∞.

$$k' = -\frac{1}{k} \tag{12}$$

Pokud tedy zobrazíme stejnou přímku v obou souřadných systémech, jsou směrnice v obou souřadných systémech na sebe kolmé. Hodnota jdoucí k nekonečnu je zároveň převedena na hodnotu jdoucí k nule.

Nevýhoda této realizace je zřejmá – nutnost dvou různých popisů a ve výsledku dvou různých algoritmů pro zjištění směru pohybu v závislosti na typu momentálně používaného souřadného systému.

K detekci bodů, které náležejí přímce, byl použit algoritmus, který pouze počítal průměrnou polohu všech bílých bodů v řádku nebo sloupci. Z detekovaných bodů pak byly vypočteny parametry *k* a *b* pomocí metody nejmenších čtverců. Takto jednoduchá metoda je dostačující, pokud uvažujeme pouze přímku nebo zatáčku, ale je zřejmé že snadno selže v případech, kdy jsou v obrazech i jiné objekty anebo v případě průjezdu křižovatkou.



Obrázek 18: Původní algoritmus detekce čáry

Tento způsob popisu a detekce trajektorie bylo nutné nahradit, protože nevyhovoval jak z hlediska způsobu reprezentace trajektorie, tak z hlediska robustnosti. Nový algoritmus, který byl použit při praktických testech, je popsán v následující kapitole.

6.2 DETEKCE A REPREZENTACE ZVÝRAZNĚNÉ TRAJEKTORIE

Jak bylo řečeno v minulé kapitole, je popis přímky rovnicí (11) nevhodný. Proto byla vzhledem k řešené úloze použita rovnice přímky ve tvaru:

$$r = x\cos\theta + y\sin\theta \tag{13}$$

Už z této volby reprezentace plyne, že pro detekci trajektorie může být použita Houghova transformace.

Výhodou této transformace je odolnost proti zašumění vstupního obrazu a přesná detekce trajektorie pokud bude její hrana (nebo prahovaný obraz) v obraze nejvíce odpovídat přímce. Tímto je možno detekovat trajektorii i v případě, že se v obraze vyskytují jiné objekty, které by mohly narušit její detekci.

Nevýhodu Houghovy transformace je výpočetní náročnost – pro každý bod obrazu, který je potenciální přímkou, musí vypočíst hodnotu r pro každé θ z intervalu

<0,360). Pokud bychom tuto transformaci aplikovali na prahovaný obraz, kde je poměrně velké množství bodů náležejících objektu (trajektorii), byl by algoritmus příliš zdlouhavý a pomalý. Zrychlení transformace bylo provedeno dvěma způsoby:

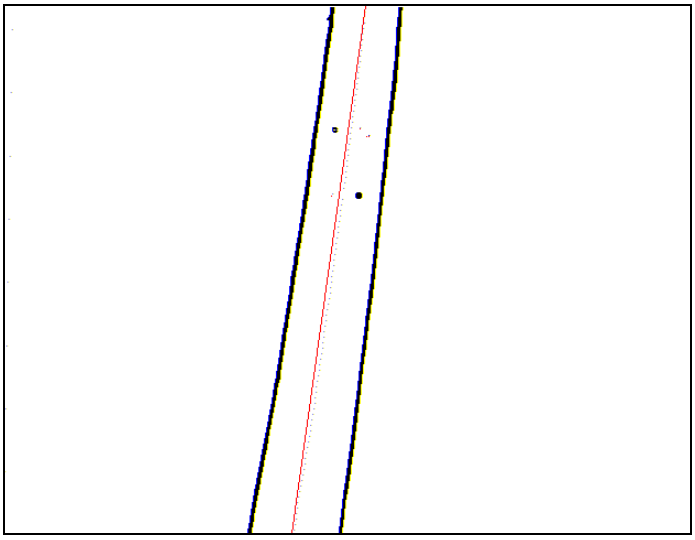
1. Byla zmenšena množina hodnot, kterých může nabývat r a θ. Konkrétně byly uvažovány pouze kladné hodnoty r (od 0 do velikosti úhlopříčky obrazu s krokem 5) a úhel θ nabýval hodnot <0,360) s krokem 3°.
2. Byl snížen počet bodů pro výpočet tak, že byly označeny středy potencionálních trajektorií v závislosti na znalosti minimální a maximální šířky této trajektorie v obraze s určitou tolerancí.

Tímto přístupem bylo dosaženo značné redukce výpočetní náročnosti.

Algoritmus označení středů čáry

1. Zjistíme nebo určíme minimální a maximální šířku čáry s ohledem na vlastnosti snímané scény.
2. Obraz procházíme po sloupcích a řádcích.
3. Nalezneme-li počátek objektu (změna jasu obrazu z 0 na 255) hledáme mezi minimální a maximální hodnotou konec objektu (změna obrazu z 255 na 0), nalezneme-li, označíme střed mezi těmito body hodnotou jasu 128.
4. Pokračujeme v procházení obrazu, dokud neprojdeme celý obraz.

Pro hodnotu objektu byla v našem případě použita hodnota jasu 255, pro pozadí 0, střed čáry byl označen hodnotou jasu 128. Výsledek je na obrázku 19 nalezené body jsou označeny červenou barvou pro lepší viditelnost.



Obrázek 19: Nalezení středu čáry

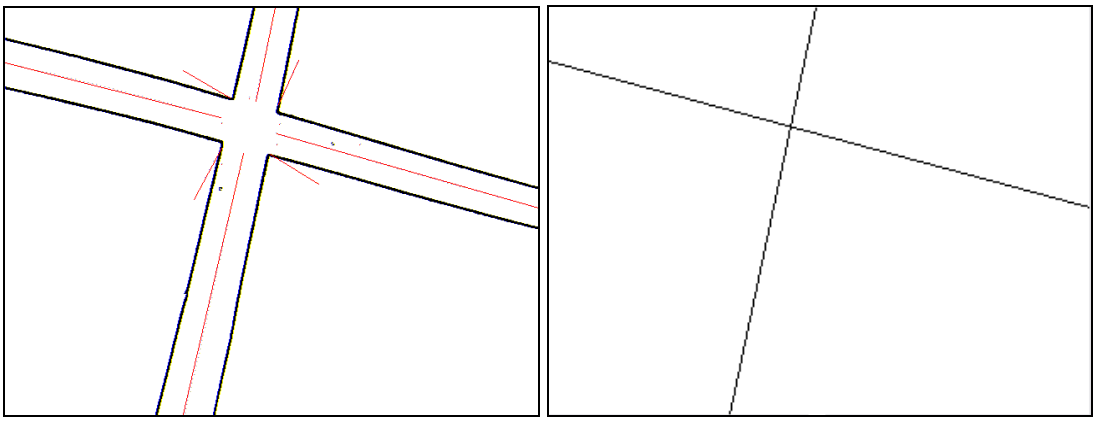
Na takto označené středy čar je použita Houghova transformace která díky tomu pracuje se silně zredukovaným počtem bodů. Následující algoritmus je upraven pro potřeby hledání dvou přímek:

Houghova transformace

1. Nadefinujeme tabulku parametrů o rozměrech definovaných počtem hodnot r a θ. Všechny počáteční hodnoty jsou nulové.
2. Procházíme obraz bod po bodu.
3. Nalezneme-li bod patřící objektu (vzhledem k předcházejícímu algoritmu hodnotu jasu 128), spočteme všechny hodnoty r pro všechna θ podle vztahu (13), na odpovídajících pozicích tabulky parametrů zvýšíme hodnotu o jedna.
4. Opakujeme, dokud neprojdeme všechny body obrazu.
5. Hledáme maximum *Max1* hodnoty v tabulce parametrů, po jeho nalezení získáme parametry první přímky (r_1,θ_1).
6. Vynulujeme určité okolí *O* maxima *Max1*.
7. Hledáme další maximum *Max2*, které má hodnotu nejméně *k*Max1*, pokud jej nalezneme, máme parametry druhé přímky (r_2,θ_2), pokud ne, druhá přímka v obrazu neexistuje.
8. Algoritmus vrátí maxima obou přímek a jejich parametry.

Vynulování okolí v kroku 6 zajišťuje, že druhé maximu nebude nalezeno pro přímku, která by byla velmi podobná první přímce, protože by se nejspíše jednalo o přímku nalezenou z bodů stejné čáry. Hodnota *k* určuje, jak blízká musí být hodnota *Max2* hodnotě *Max1*, aby bylo rozhodnuto, že se skutečně jedná o přímku, kterou chceme nalézt.

Druhá přímka je hledána z důvodu nutnosti řešení křižovatek nebo zatáček. Pokud algoritmus vrátí dvě přímky, dochází, je-li to možné, ještě k jejich rozdělení na přímku více rovnoběžnou (*P1*) se směrem pohybu robota a méně rovnoběžnou se směrem pohybu robota (*P2*). Důvodem je úvaha, že pokud existují v obraze 2 přímky, pak přímka rovnoběžná s pohybem robota je přímkou, po které se zřejmě momentálně pohybuji, zatímco přímka více kolmá na směr pohybu robota je přímkou, (tvořící například potenciální křižovatku) na kterou je možné odbočit.



Obrázek 20: Přímky získané Houghovou trasformací

Na obrázku 20 je vlevo vidět prahovaný obraz a vpravo již přímky vypočtené z rovnic získaných Houghovou transformací. Nalezené potenciální středy přímek jsou opět označeny červenou barvou. Jako *P1* bude označena svislá přímka, jako *P2* bude označena horizontální přímka. Z pohledu robota se totiž nacházíme ve středu dolní části obrazu a obraz, který vidíme, se nachází před robotem.

6.3 DETEKCE ZVÝRAZNĚNÉ TRAJEKTORIE - SHRNUTÍ

Tímto krokem získáváme matematický popis opticky zvýrazněné trajektorie, se kterým již můžeme pracovat při určování směru pohybu robota. Tento algoritmus ve spojení s mícháním barevných kanálů je velice robustní. Abychom dostali nesprávnou přímku, musel by se v obraze nacházet objekt stejné barvy, podobné šířky jako čára, musel by být rovný a navíc by musel mít v obraze dostatečnou délku, aby prošel kritériem *k*Max1*.

7. ZAJIŠTĚNÍ NAVIGACE ROBOTA

S pomocí matematické reprezentace z minulého kroku je již možné přistoupit k tvorbě algoritmů pro určení směru pohybu robota. Jako výstup algoritmů byla zvolena odchylka od směru, kterým se chceme pohybovat ve stupních.

I když dále v algoritmech není uvedeno, byla v určení směru vytvořena i podmínka pro zastavení robota v případě, že v obraze neexistuje přímka (hodnoty *Max1* a *Max2* jsou nižší než nejnižší určená hodnota), aby bylo chování robota definováno i v případě ztráty cesty nebo konce čáry.

7.1 NAVIGACE PO ČÁŘE

Algoritmus použitý k navigaci robota je uveden níže, jeho výstupem je úhel, o který je robot odchýlen od požadovaného bodu, ke kterému má směřovat. Jako bod, ke kterému robot neustále směřuje, byla vybrána poslední třetina detekované přímky v obraze.

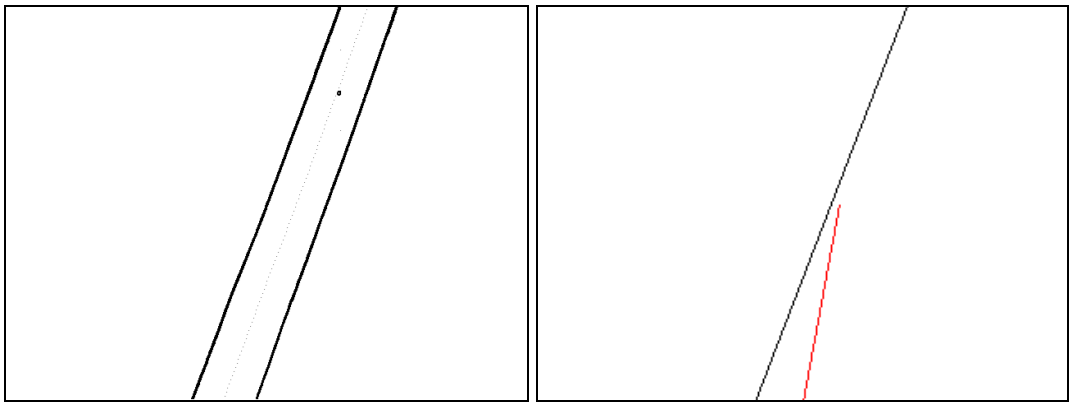
Pokud se v obraze nenachází detekovatelná čára, je robot zastaven. V případě jedné přímky je určován směr k počátku poslední třetiny přímky v obrazu. Pokud z předchozího algoritmu dostáváme 2 přímky, které netvoří křižovatku (nejsou na sebe přibližně kolmé), jedná se zřejmě o zatáčku (více navazujících kratších čar) a směr je určován podle té přímky, která má větší váhu, tzn. má větší hodnotu maxima v tabulce parametrů Houghovy transformace. Tím je možno řídit robota po navazujících přímkách. Robot je vždy řízen podle čáry, která má v obraze momentálně větší plochu.

Popis algoritmu řízení podle jedné detekované přímky

1. Pokud existují dvě přímky, je zvolena ta z přímek, která má větší váhu. Pokud ne, použijeme aktuální přímku.
2. Je nalezen začátek poslední třetiny přímky (poslední třetinou se myslí třetina přímky nejvíce vzdálená od počátku obrazu), tedy bod, ke kterému chceme směřovat.
3. Je vypočtena a vrácena odchylka od požadovaného směru ve stupních.

Kladný úhel znamená otáčení robota doprava, záporný doleva. Pokud je vrácena 0 směřujeme správným směrem. Směřování k poslední třetině bylo vybráno z toho důvodu, že kdybychom se dostali do situace, kdy je přímka kolmo ke směru pohybu robota, znamenalo by směřování například ke středu přímky přejetí a ztrátu čáry.

Na obrázku 21 je vidět výsledek určení směru, algoritmus vrátil hodnotu 11° , tedy že by robot měl změnit směr pohybu o 11° vpravo, aby se dostal k cílovému bodu. (pozn.: detekovaná čára má černou barvu, červená přímka ukazuje směr k poslední třetině přímky)



Obrázek 21: Určení směru podle jedné přímky

Při testech na platformě UTAR nebyly s tímto určováním směru žádné problémy, robot byl schopen sledovat i čáru poměrně často a prudce měnící směr. Tento algoritmus určení směru je také důvod, proč není výhodné, aby robot sledoval scénu daleko dopředu, docházelo by totiž k příliš brzké změně směru v závislosti na směřování čáry daleko před robotem.

7.2 NAVIGACE NA KŘIŽOVATCE

Při navigaci na křižovatce ještě předtím, než je nutné odbočit, známe směr, kterým se na křižovatce vydáme (je určen předem nebo zjištěn pomocí značky křižovatky jak je uvedeno v následující kapitole).

Křižovatka je definována kolmostí dvou přímek v obraze, kolmost přímek je rozpoznávána s určitou tolerancí vzhledem ke geometrickému zkreslení obrazu a přesnosti určení parametrů přímek. Při testech byla kolmost přímek určována s tolerancí ±15°. Pokud jsou přímky v obraze kolmé, je detekována křižovatka.

Na křižovatce je možno pokračovat rovně, odbočit vpravo, vlevo nebo zastavit. Zastavení má význam například při označení dveří, u kterých má robot počkat (počkat na instrukce, doručit poštu,…).

Protože značení křižovatek popsané v následující kapitole pracuje s kódem uvedeným přímo na křižovatce, je nutné počkat se změnou směru a dát algoritmu čas k přečtení kódu křižovatky a určení směru. Proto ke změně směru dochází až ve chvíli, kdy je střed křižovatky za určitou definovanou hranicí. Poloha středu křižovatky je vypočtena jako průsečík přímek.

Popis algoritmu

1. Běží algoritmus popsaný v předchozí kapitole, dokud není detekována křižovatka.
2. Po detekci křižovatky čekáme se změnou směru, dokud se střed křižovatky nenachází v definovaném místě.
3. Další směr pohybu je určen podle parametru, který je předán funkci pro křižovatku, pokud má robot pokračovat rovně, řídíme jej podle přímky *P1* pokud má robot pokračovat vlevo nebo vpravo, řídíme jej podle přímky *P2* příslušným směrem, pokud má robot zastavit, zastavíme.
4. Když po odbočení, či projetí křižovatky, křižovatka zmizí z obrazu, je řízení opět předáno do bodu 1.

Obrázek 22: Odbočení na křižovatce

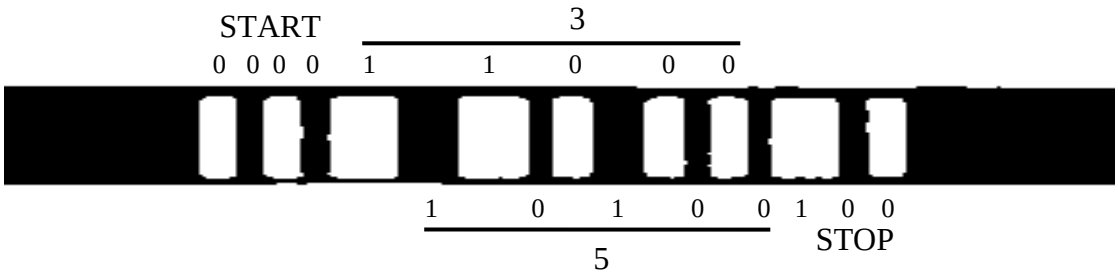
7.3 ZNAČENÍ KŘIŽOVATEK

Protože je při odbočování nutné vědět, na které křižovatce se nacházíme, bylo vytvořeno značení křižovatek pomocí čárového kódu. Tento kód nenese přímo informaci o směru, kterým se má robot vydat, ale označuje číslem křižovatku. Pomocí tohoto čísla je možno určit pozici robota a směr, kterým by se měl dále vydat pomocí dat uložených v paměti. Na takto označených křižovatkách by bylo možno postavit celou mapu a používat ji k hledání a generování cesty mezi dvěma body. Tvorba mapy a hledání cesty však již přesahuje zadání a rozsah této práce. Zde je uvedena pouze metoda značení křižovatek.

K označení křižovatek byl použit prokládaný čárový kód v kódování 2 z 5. Tabulka kódování hodnot je uvedena níže. Poměr délek znaků 1 a 0 bývá 2:1 nebo 3:1. Kód 2 z 5 má poměrně dobrou odolnost proti chybám při čtení, prokládáním navíc bez přílišného nárůstu délky kódu umožňuje uložit dvakrát více hodnot. V kódu je umístěn start a stop znak umožňující identifikovat počátek a konec kódu.(Wikipedia [5])

Znak	1.čára	2.čára	3.čára	4.čára	5.čára
0	0	0	1	1	0
1	1	0	0	0	1
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	1
5	1	0	1	0	0
6	1	1	0	0	0
7	0	0	0	1	1
8	1	0	0	1	0
9	0	1	0	1	0
start	0	0	0	0	
stop	1	0	0		

Tabulka 1: Kódování čárového kódu 2 z 5



Obrázek 23: Čárový kód v prahovaném obraze

Vzhledem k rozmazávání obrazu pohybem byl kód umístěn na křižovatce na čáře kolmé na pohyb robota, čímž byla zaručena jeho čitelnost. K označení křižovatky jsou tedy potřeba dva kódy s ohledem na směr, ze kterého robot přijíždí.

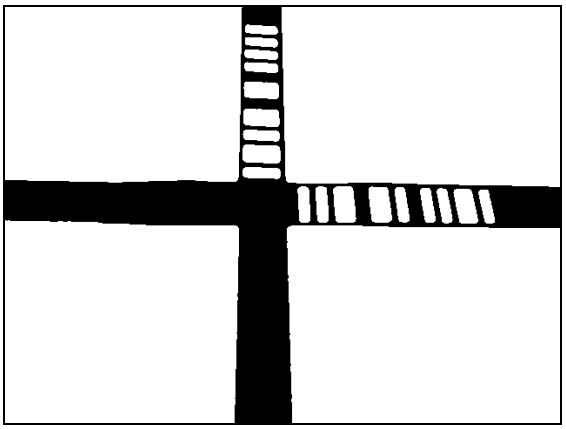
Ke čtení čárového kódu byla využita znalost polohy čáry. Při detekování křižovatky je procházena *P2* kolmá na pohyb robota bod po bodu a jsou ukládány délky jednotlivých segmentů kódu. Tyto uložené délky jsou rozděleny podle délky na znaky 0 a 1 poté dekódovány. Výsledkem je číslo označující křižovatku. Délky segmentů jsou určovány v pixelech.

Algoritmus čtení kódu

1. Bod po bodu procházíme přímku kolmou na pohyb robota (*P2*).
2. Ukládáme délky jednotlivých segmentů kódu (segmenty jsou odděleny změnou z hodnoty jasu z 0 na 255 a zpět nebo opačně – viz obrázek 23).
3. Je určena hranice délky mezi délkou segmentu kódujícím znak 1 a 0. Poté jsou délky převedeny na binární kód.
4. Pokud je první detekován znak stop, je kód převrácen.
5. Je dekódováno číslo nesené kódem.
6. Pokud se nepodaří správně přečíst celý kód (chyba v délce, chybí start/stop znak, binární kód neodpovídá žádnému kódování v tabulce,…) je čtení vyhodnoceno jako chybné a zopakováno v příštím snímku.

Největším problémem je krok 3. Určení délky segmentu musí být provedeno až při čtení, protože nemůžeme předem vědět, jak dlouhý bude kód v obraze. Navíc může být obraz špatně segmentován nebo může být porušená čára. Proto byly ve skutečnosti kroky 3 až 5 opakovány pro všechny délky segmentů tak, že délka každého segmentu byla postupně použita jako délka znaku 0 a poté se testovalo, jestli výsledkem bude použitelný kód odpovídající délkou, označením start/stop a kódováním. Tím byla zvýšena úspěšnost čtení kódu a zaručeno čtení i při porušení čáry nebo chybě v segmentaci.

Tento algoritmus je funkční pro prahovaný obraz, pro obraz hran je nutné jej mírně upravit, aby nezapočítával samotné hrany jako segmenty.



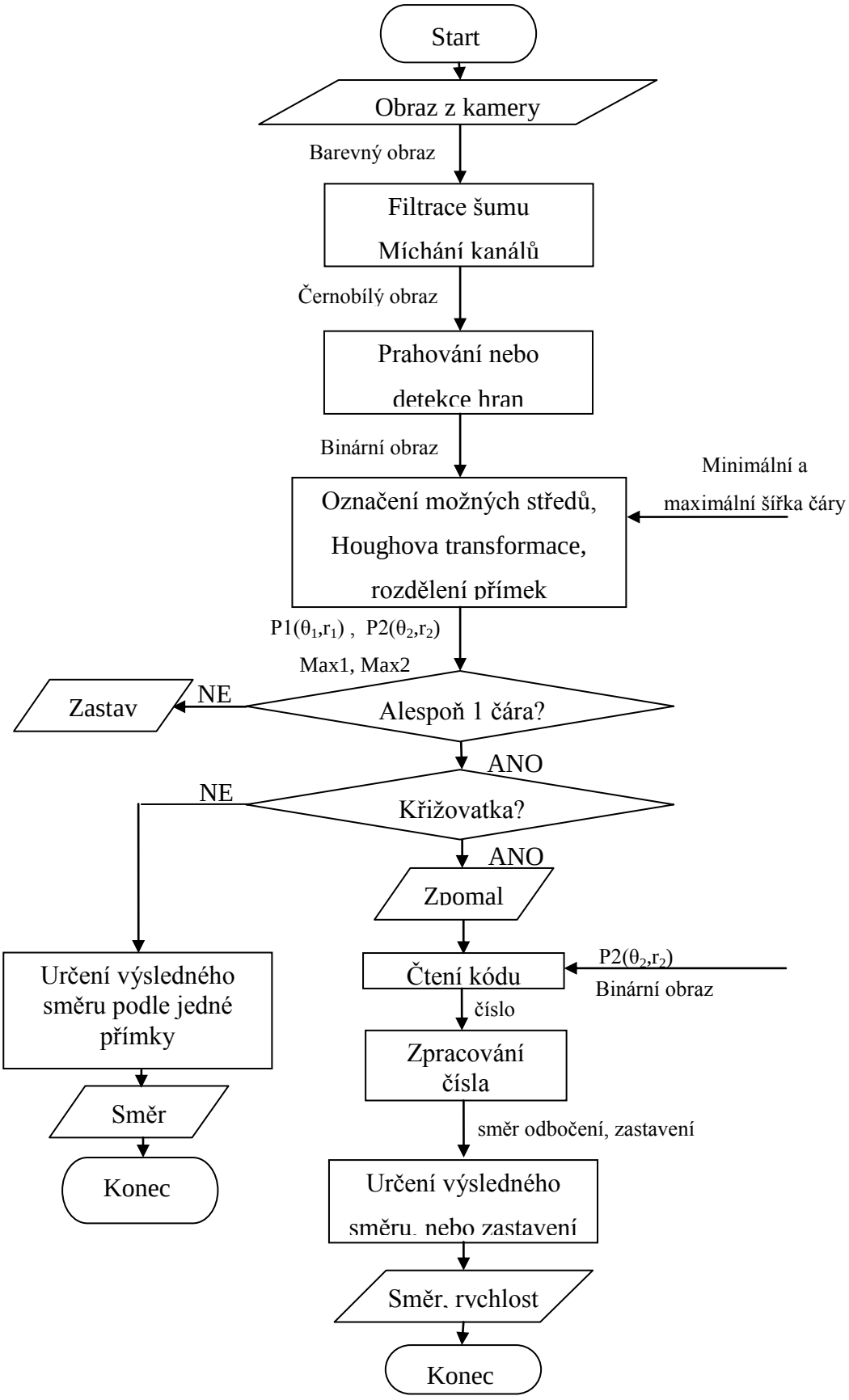
Obrázek 24: Umístění kódů křižovatky

Na obrázku 24 je umístění kódů na křižovatce. Robot přečte kód umístěný na čáře kolmé ke směru pohybu a v souvislosti s významem, který pro něj tento kód má, odbočí, projede nebo zastaví na křižovatce.

8. VÝSLEDNÝ ALGORITMUS

Na následujícím obrázku jsou shrnuty všechny kroky od předzpracování obrazu po určení dalšího směru pohybu robota. Hodnota minimální a maximální šířky čáry je zjištěna při kalibraci na testovacím snímku. Zpomalení po detekci křižovatky je implementováno z důvodu snížení rozmazávání obrazu a zlepšení čtení kódu křižovatky.

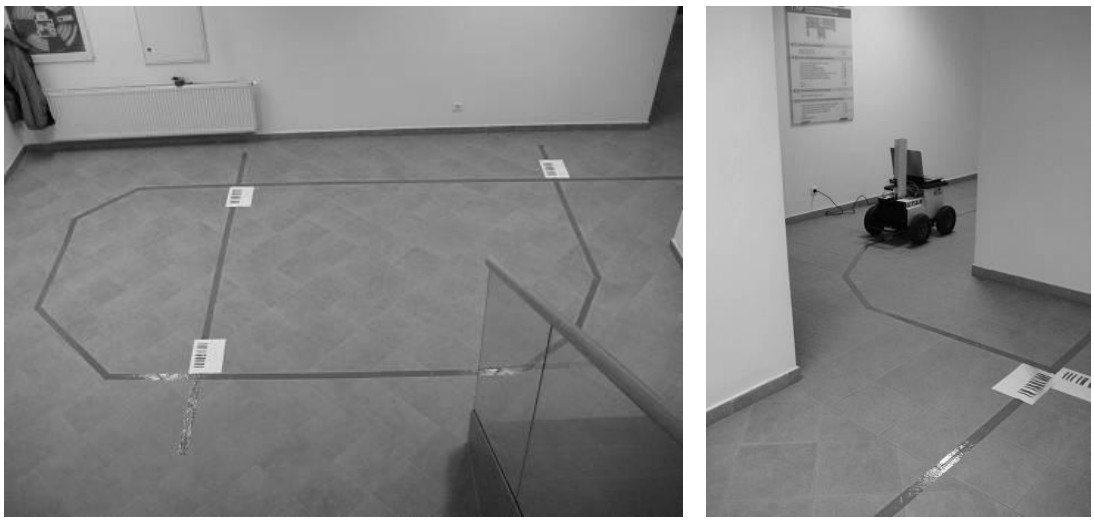
Diagram odpovídá zpracování obrazu, které probíhá ve vytvořené testovací aplikaci. Algoritmus v programu proběhne pro každý nově získaný snímek. Výstupem algoritmu je směr a popřípadě rychlost, kterou by se měl robot pohybovat.



Obrázek 25: Výsledný algoritmus

9. TESTOVÁNÍ

Testování algoritmu proběhlo na robotu UTAR s kolovým podvozkem. Algoritmus byl během odlaďování testován několikrát. Finální testovací dráha sestávala ze 4 křižovatek a několika zatáček. Čára byla nalepena zelenou páskou, značení křižovatek bylo provedeno překrytím čáry papírovou maskou. Velikost kódu byla přibližně na velikost stránky formátu A4. Během testování byl vyzkoušen i vliv poškození čáry a umístění jiných objektů v zorném poli robota na jeho chování.



Obrázek 26: Testovací dráha

K získávání obrazu byla použita obyčejná web kamera umístěná v přední části robota a připojená přes USB port k notebooku, na kterém běžel program zpracovávající obraz pomocí výše uvedeného algoritmu. Použitá kamera byla webová kamera (Logitech QuickCam® Pro 9000), bohužel výrobce neuvádí zorný úhel ani další parametry kamery kromě rozlišení (až 1600x1200 bodů), funkce automatického ostření obrazu a snímkovací frekvence (až 30 snímků za sekundu). Obraz byl pořizován v rozlišení 640x480 při 10 snímcích za sekundu. Snímkovací frekvence byla volena s ohledem na čas nutný ke zpracování obrazu. Kamera byla umístěna ve výšce asi 40cm s úhlem pohledu přibližně 45°. Výška a natočení kamery bylo voleno s ohledem na velikost čáry v obraze a čitelnost kódu. Notebook byl propojen s podvozkem robota pomocí rozhraní RS232. Použití notebooku bylo

výhodné z hlediska testování a odlaďování, finální implementace by představovala nahrání kódu do PC robota.

Obrázek 27: Robot UTAR

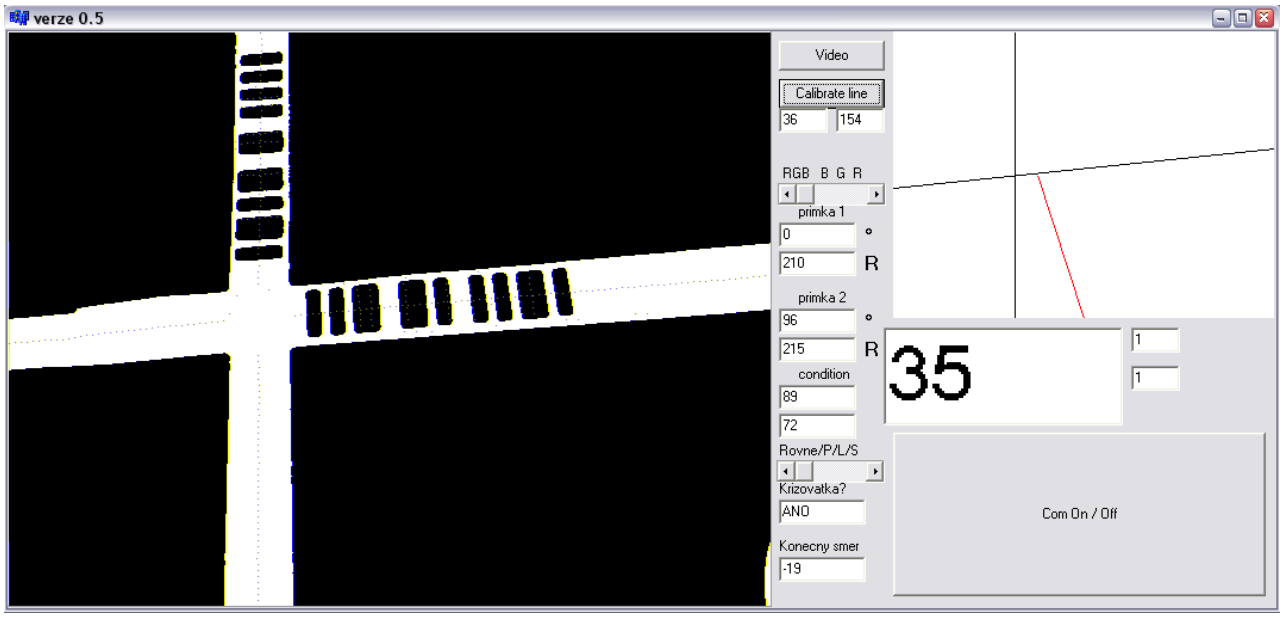
Pomocí algoritmu byl řízen přímo podvozek přes rozhraní RS232 – funkci pro řízení podvozku byl předáván směr otáčení a rychlost. Směr otáčení byl závislý na znaménku, rychlost otáčení robota na velikosti úhlu vráceného algoritmem, rychlost byla nastavena pevně a měněna při detekci křižovatky.

Pokud vnímáme čáru jako trajektorii, po které se má robot pohybovat, pak kamera zajišťuje zpětnou vazbu a výstup algoritmu lze vnímat jako odchylku od správného směru. Rychlost reakce na tuto odchylku byla dána závislostí mezi velkostí úhlu a rychlostí otáčení robota. Jelikož cílem této práce není přímo řízení podvozku, ale zpracování obrazu, byla závislost mezi úhlem a rychlostí otáčení robota nastavena empiricky s ohledem na jeho chování a rychlost reakce při sledování čáry.

9.1 PROGRAM PRO ZPRACOVÁNÍ OBRAZU

Pro účely testování a vzhledem k zadání byla vytvořena aplikace v jazyce C++. Algoritmus zpracování obrazu v této aplikaci je shodný s algoritmem uvedeným v kapitole 8. Program a zdrojové kódy jsou přílohou práce.

Pro získání obrazu z webové kamery bylo využito win32 API. Program obsahuje také kód pro komunikaci s podvozkem přes sériovou linku RS232 a je vybaven grafickým rozhraním pro usnadnění ovládání a odlaďování algoritmu.



Obrázek 28: Okno aplikace

V grafickém rozhraní je zobrazen zpracovaný obraz a vykresleny přímky, které jsou výsledkem Houghovy transformace. Jsou zde také uvedeny informace o parametrech jednotlivých přímek, detekci křižovatky, směru a číslu křižovatky.

Tlačítka:

- Video – vyvolá nabídku s volbou snímacího zařízení a jeho parametrů, je možno nastavit i rozlišení, program byl však testován pouze pro rozlišení 640x480, pro jiná rozlišení obrazu není zaručena korektní funkčnost.
- Calibrate line – slouží ke kalibraci šířky čáry z obrazu, tato funkce vyžaduje správně segmentovaný obraz čáry bez dalších objektů.
- Com On/Off – vypínání a zapínání komunikaci, při zastavení komunikace dojde nejdříve k zastavení robota a teprve potom je komunikace ukončena. Číslo portu je v programu nastaveno pevně na COM2.

- Posuvníky – posuvník výše slouží k nastavení barevného kanálu, který bude použit pro zpracování. Druhý posuvník slouží k ručnímu zadání směru na příští křižovatce.

Během testování nebyl směr zadáván ručně. Čísla jednotlivých křižovatek byly spolu s odpovídajícími směry uloženy v datové struktuře. Program tuto strukturu používal k určení směru podle čísla křižovatky, na které se nacházel. Robot byl při všech testech řízen touto aplikací.

9.2 POZNATKY Z TESTOVÁNÍ

Algoritmy byly testovány několikrát a podle dosažených výsledků postupně upravovány. Hlavní poznatky jsou uvedeny v následujících odstavcích.

9.2.1 Sledovaná trajektorie

Během testu byla používána obyčejná černá nebo barevná lepicí páska. Toto řešení není ideální hlavně vzhledem k jejímu lesklému povrchu. Odlesky mohou způsobovat chyby v segmentaci a čtení čárového kódu. Mnohem lepší je použití matné pásky pro vyznačení cesty.

9.2.2 Získání a zpracování obrazu

Slabým místem, zjištěným při testování, byla kvalita obrazu použité kamery. Obraz byl sice ostrý, s nízkou úrovní šumu, ale při pohybu docházelo k jeho rozmazání. Rozmazání obrazu bylo tak velké, že bylo nutno použít prahovaný obraz místo obrazu hran. Také bylo nutno změnit způsob značení křižovatek tak, aby nedocházelo k rozmazání jednotlivých segmentů kódu – původní myšlenkou bylo umístit kód na čáru ve směru pohybu robota, aby byl přečten ještě před křižovatkou. Z těchto problémů jasně vyplývá, že je nutno věnovat velkou pozornost volbě použitých zařízení pro získání obrazu.

9.2.3 Detekce trajektorie

Původní algoritmus uvedený v kapitole 7.1 nebyl testován vůbec. K testování byla používána pouze metoda založena na Houghově transformaci. Původně byla uvažována černá čára na bílém podkladu. Velice brzy se však ukázalo, že není algoritmicky možné eliminovat všechny vlivy prostředí – mnoho objektů v okolí

robota mělo vlastnosti podobné čáře, včetně rozměrů, což vedlo k chybám v určování směru. Situace se velmi zlepšila použitím barevné čáry a jejím zvýrazněním pomocí míchání barevných kanálů, čímž došlo k eliminaci vlivu většiny objektů v zorném poli robota. Během testování byla čára přerušena nebo byly na ní a kolem ní umístěny různé objekty, směr byl však vždy určován správně. (kromě případů, kdy čára úplně zmizela z obrazu – došlo k zastavení robota)

9.2.4 Určování směru pohybu

S algoritmem určování směru, detekce křižovatek a odbočením nebyly problémy během celého testování.

9.2.5 Čtení kódů křižovatek

Čárový kód použitý k označení křižovatek je poměrně dobrým principem značení, díky umístění přímo na čáře bylo čtení kódu poměrně lehké. Jako nevýhoda během testování se ukázala délka kódu – k spolehlivému čtení je nutné, aby měl každý segment v obraze dostatečnou šířku, obzvláště pokud dochází k rozmazávání obrazu pohybem robota. Řešením by mohlo být použití lepší kamery a vyššího rozlišení nebo použití kratšího kódu. Během finálního testování byl tento problém vyřešen použitím co největšího kódu a zpomalením pokud robot detekoval křižovatku. Robot celou testovací dráhu absolvoval mnohokrát, k chybě přečtení kódu křižovatky došlo pouze jednou.

10. ZÁVĚR

Cílem bylo základní seznámení s problematikou počítačového vidění a jeho aplikace na konkrétní úlohu, jejíž řešení je hlavní náplní práce.

V práci byl prezentován úvod do problematiky počítačového vidění a návrh postupu pro řešení úlohy. Jednotlivé algoritmy pro předzpracování, segmentaci, detekci čáry a její matematickou reprezentaci a čtení čárového kódu vycházejí z počítačového vidění. Algoritmy pro určování směru, odbočení a rozpoznávání křižovatek jsou výsledkem využití informací získaných počítačovým viděním. Kompletací jednotlivých částí řešení vznikl algoritmus schopný navigovat robota po zvýrazněné trajektorii, rozpoznat křižovatky a odbočit na nich.

Výsledný algoritmus byl několikrát prověřen testováním na modelové situaci a vykazoval dobrou spolehlivost. Největší problémy při rozpoznání čáry vznikly použitou kamerou a zpočátku použitím černé čáry a černobílého obrazu, to však bylo vyřešeno použitím barevné čáry v kombinaci s vhodným předzpracováním obrazu. Poté již detekce čáry pracoval bez problémů. Čtení kódu je problematické z hlediska velikosti kódu v obrazu, i přesto bylo při čtení dosaženo poměrně dobré spolehlivosti.

Dalším logickým krokem by bylo vytvoření vhodné mapy a algoritmů pro její tvorbu a hledání nejkratší cesty. Vytvoření mapy však již není součástí zadání a nesouvisí s počítačovým viděním.

11. LITERATURA

[1] HLAVÁČ, V., ŠONKA, M.: Počítačové vidění. Praha: Grada, 1992.

[2] HOŘIČKA, J.: Počítačové zpracování digitálních obrázků - Houghova transformace. [online] K7 – vědecko populární časopis Fakulty mechatroniky TU v Liberci. 04/2005. Dostupné z www: <http://k7.vslib.cz/files/k7_05_4.pdf>. ISSN 1214-7370.

[3] MUDROVÁ, Martina.: Geometrické transformace obrazu a související témata. 2004.[online]. Dostupné z www: <http://uprt.vscht.cz/ucebnice/ZOB/prednasky/09-TRANSFORMACE/transformace-tisk.pdf>

[4] Wikipedia The Free Encyklopedia.: Thresholding (image processing).[online]. Dostupné z www: <http://en.wikipedia.org/wiki/Thresholding_%28image_processing%29>

[5] Wikipedia The Free Encyklopedia.: Interleaved 2 of 5.[online]. Dostupné z www: < http://en.wikipedia.org/wiki/Interleaved_2_of_5>